Influence of Tree Topology Restrictions on the Complexity of Haplotyping with Missing Data[*]

Michael Elberfeld
Institut für Theoretische Informatik
Universität zu Lübeck
D-23538 Lübeck, Germany
elberfeld@tcs.uni-luebeck.de

Ilka Schnoor
Institut für Informatik
Christian-Albrechts-Universität zu Kiel
D-24118 Kiel, Germany
ils@informatik.uni-kiel.de

Till Tantau
Institut für Theoretische Informatik
Universität zu Lübeck
D-23538 Lübeck, Germany
tantau@tcs.uni-luebeck.de

January 23, 2012

Abstract

Haplotyping, also known as haplotype phase prediction, is the problem of predicting likely haplotypes based on genotype data. One fast haplotyping method is based on an evolutionary model where a perfect phylogenetic tree is sought that explains the observed data. Unfortunately, when data entries are missing, which is often the case in laboratory data, the resulting formal problem IPPH, which stands for incomplete perfect phylogeny haplotyping, is NP-complete. Even radically simplified versions, such as the restriction to phylogenetic trees consisting of just two directed paths from a given root, are still NP-complete; but here, at least, a fixed-parameter algorithm is known. Such drastic and ad hoc simplifications turn out to be unnecessary to make IPPH tractable: we present the first theoretical analysis of a parametrized algorithm, which we develop in the course of the paper, that works for arbitrary instances of IPPH. This tractability result is optimal insofar as we prove IPPH to be NP-complete whenever any of the parameters we consider is not fixed, but part of the input.

Classification: computational biology, computational complexity, fixed-parameter algorithms, haplotyping, phylogenies

1 Introduction

Haplotype phase prediction is a preprocessing step in genomic disease and medical condition association studies. In these studies two groups of people are considered, where one group has a certain disease or medical condition while the other has not, and one tries to find correlations between group membership and the genomic data of the individuals in the groups. The genomic data typically consists of information about which bases are present in an individual's DNA at so-called SNP sites (single nucleotide polymorphism sites). While the DNA sequences of different individuals are mostly identical, at SNP sites there may be variations. Low-priced methods for large-scale inference of genomic data can read out, separately for each SNP site, the bases present, of which there can be two since we inherit one chromosome from our father and one from our mother. However, since the bases at different sites

[*] This is the accepted author manuscript of the article http://dx.doi.org/10.1016/j.tcs.2012.01.015. A preliminary version of this paper was presented at the TAMC 2009 conference [8].

are determined independently, we have no information to which chromosome a base belongs to. For *homozygous sites,* where the same base is present on both chromosomes, this is not a problem, but for *heterozygous sites* this information, called the *phase* of a SNP site, is needed for accurate correlations. The idea behind *haplotype phase prediction* or just *haplotyping* is to predict likely phases computationally based on the laboratory data (which misses this information). For an individual, the genomic input data without phase information is called the *genotype* while the two predicted chromosomes are called *haplotypes*.

From a mathematical point of view, haplotypes can conveniently be coded as strings over the alphabet $\{0,1\}$, where for a given site 0 stands for one of the bases that can be observed in practice, while 1 encodes a second base that can also be observed. (The case that three bases are observed happens so seldom that it can be ignored.) A genotype g is, conceptually, a sequence of sets that arises from two haplotypes h_1 and h_2 as follows: The ith set in the sequence g is $\{h_1[i], h_2[i]\}$. However, it is customary to encode the set $\{0\}$ as 0, to encode $\{1\}$ as 1, and $\{0,1\}$ as 2, so that a genotype is actually a string over the alphabet $\{0,1,2\}$. For example, the two haplotypes 0110 and 0101 give rise to (we also say *explain*) the genotype 0122; and so do 0100 and 0111.

Since different haplotype pairs can explain the same genotype and any single haplotype is equally likely *a priori,* haplotyping is not possible if only a single genotype is given. However, for a whole set of genotypes from a larger group of different individuals, certain sets of haplotypes that explain these genotypes are more likely than others. For instance, a small set of explaining haplotypes is more likely than a large set since haplotypes mutate only rarely. One important method of haplotyping is based on the *perfect phylogeny approach* proposed by Gusfield [15]. The underlying idea is that the probability of a mutation in a single SNP site is so small that we may safely assume that another (backward) mutation at the same site will not have happened. Accordingly we seek a set of haplotypes that can be arranged in a tree in which every edge marks a mutation of some SNP site. This means that the edge connects two components of the phylogenetic tree where in one component all haplotypes have a 0-entry at this site and in the other component all haplotypes have a 1-entry there. A tree with this property is called a *perfect phylogeny* and the computational question of whether a given set of genotypes can be explained by haplotypes that can be arranged in a perfect phylogeny is the *perfect phylogeny haplotyping problem* (PPH).

Numerous results on the complexity of PPH and its variants are known. Gusfield showed that the problem can be solved in polynomial time [15], further papers first presented simpler polynomial-time algorithms [1, 10] and later even linear-time algorithms [3, 5, 18, 20]. In [7] the first author showed that PPH is complete for logarithmic space. However, in practice, laboratory data is never perfect and some entries may be missing in the input genotypes. In this case, the input matrices may contain ?-entries in addition to the 0-, 1-, and 2-entries. The objective is then to replace the missing entries by normal entries such that the resulting genotypes are elements of PPH. This problem is known as IPPH, where the I stands for *incomplete*. Unfortunately, IPPH is NP-complete [23]. A heuristic is known for solving it [21], but no guarantees can be made concerning its runtime.

The problems PPH and IPPH both have a *directed* variant. In real data, some genotype is typically completely known and is completely homozygous, which means that one of the sought explaining haplotypes is already known. This problem variant is called "directed" because the position of the known haplotype in the phylogenetic tree singles out a node, which is then regarded as the root and gives an orientation to the tree. The resulting problems are called DPPH and IDPPH, with D standing for "directed." Although directedness is a simplification, IDPPH is still NP-complete [17].

In order to tackle the complexity of IPPH, one possible approach is to study its fixed-parameter tractability. The idea behind the framework of fixed-parameter complexity, initially developed by Downey and Fellows [6], is that many NP-complete problems can in fact be solved efficiently when we allow only input instances for which a certain problem-specific parameter is small. We consider two parameters of instances for IPPH: The first parameter is the maximum number of missing entries at any single SNP site in the input. We expect this number to be small in real data – otherwise the relevance of the data would be questionable – and this expectation is backed by real genotype data, see [13] for a detailed analysis. The second parameter does not regard the input, but the topology of the phylogenies that are sought: we restrict the number of leaves in an explaining phylogenetic tree. This generalizes the concept of path phylogenies from Gramm et al. [13] which are phylogenies with only two leaves. The study of path phylogenies is motivated by the discovery that the human genome contains many so-called yin-yang haplotypes, see [24], which enforce explaining phylogenies to be path phylogenies, see [13] for details. In practice, path phylogenies are, indeed, common in the human genome, but it also happens that the underlying phylogenies have more than just two leaves.

Our Contributions. We present an algorithm that decides in time $f(k,l)n^2m^{O(l)}$, where f is an at most double-exponential function, n is the number of genotypes in the input, and m is the length of those genotypes, whether a given set of incomplete genotypes with at most k missing entries for each SNP site belongs to IPPH via a phylogeny having at most l leaves. This algorithm allows us to make formal statements about the fixed-parameter tractability of IPPH. First, IPPH lies in the class XP for the parameter pair (k,l). Second and more importantly, for each fixed $l \geq 2$ the problem $\text{IPPH}_{\text{leaves}\leq l}$ (which is IPPH restricted to instances that can be explained by a perfect phylogeny having at most l leaves) is fixed-parameter tractable with respect to the number of unknown entries per SNP site.

We show that this double parametrization is necessary, in the sense that we can not set either one of the parameters aside. Our first hardness result states that IPPH is NP-complete for inputs where the number of missing entries per column and also per row is restricted by the constant 15. Our second hardness result implies that $\text{IPPH}_{\text{leaves}\leq l}$ is NP-complete for every $l \geq 2$. In fact we show that the much more restricted problem $\text{IDPP}_{\text{leaves}\leq l}$ (given a set of incomplete haplotypes and one complete haplotype, can the missing entries be filled with regular entries such that the input haplotypes can be arranged in a perfect phylogeny with at most l leaves?) is NP-complete. To cover both variants, directed and undirected, we show all hardness results for the directed case and give the fixed-parameter algorithm for the undirected case. (Note that there is a trivial reduction from the directed to the undirected case: just make a directed instance undirected and add the root genotype to the set of input genotypes.)

These results completely settle the questions left open in [13], namely whether the fixed-parameter algorithm given there for $\text{IDPPH}_{\text{leaves}\leq 2}$ can be extended to the cases where we allow a larger number of leaves (indeed, we can), look at undirected variants (again, an extension is possible), or make no restrictions on the number of leaves at all (the problem then becomes NP-complete even for a fixed number of missing entries per site).

Methods. Our fixed-parameter algorithm generalizes Gramm et al.'s result [13] that $\text{IDPPH}_{\text{leaves}\leq 2}$ is fixed-parameter tractable. The algorithm from [13] relies strongly on Gusfield's characterization [15]: Given a set of genotypes A, a directed perfect phylogeny T for it with the all-0-haplotype as its root, and any genotype g of A, the 1-entries of g label a path from the root to some node v of T and the 2-entries of g label a path containing v. Most algorithms for PPH and its variants from the literature exploit this necessary property as follows: They first reduce the problem to the directed version DPPH and then build the phylogeny by placing columns with many 1-entries and 2-entries near to the root and columns with fewer such entries far from the root. The notions "should be placed near to the root" and "should be placed far from the root" can be quantified more precisely by using Gusfield's notion of the *leaf count* of a column [15].

When the data is incomplete, no reduction from the undirected to the directed case is known. (Indeed, IPP is NP-complete while IDPP $\in$ P.) To solve the undirected problem variant $\text{IPPH}_{\text{leaves}\leq l}$, we need a replacement for the notion of leaf count and a characterization of sets of genotypes admitting undirected perfect phylogenies. We present such a replacement, which we call the *light component size*, and also a characterization in terms of the new notion of *mutation trees*. This characterization allows us to construct phylogenies in a stepwise fashion from the "outside" of the phylogeny (columns having a small light component size) to the "inside" of the phylogeny (columns having a large light component size). In each step, we only need to remember the inner part of the partial phylogeny constructed so far, making a dynamic program feasible.

Related Work. Haplotyping methods can be split into two groups: Statistical, see [11] for a literature starting point, and combinatorial. There are two main combinatorial methods: Maximum parsimony haplotyping [4, 14] and the more recent perfect phylogeny approach that was introduced by Gusfield [15] and later explored by numerous authors [1, 3, 5, 10, 18, 20].

The idea of considering restricted tree topologies to speed up haplotyping is due to Gramm et al. [13] and was recently also investigated in the context of finding block partitions [12]. A different approach to deal with the NP-completeness of IPPH is due to Halperin and Karp [16]. They present a polynomial-time algorithm for IPPH that works for special instances satisfying the so-called "rich data hypothesis."

The influence of restricting the tree topology on the complexity of haplotyping problems has, prior to the findings of the present paper, always been benign: In [13] it is shown that $\text{IDPPH}_{\text{leaves}\leq 2}$ has a fixed-parameter algorithm. In [12] it is shown that partitioning a complete genotype matrix into a minimal number of column sets such that each set admits a perfect *path* phylogeny is equivalent, in complexity theoretic terms, to finding maximal matchings; while the same problem for arbitrary perfect phylogenies is NP-hard and even very hard to approximate. Finally, in [9] it is shown that $\text{DPPH}_{\text{leaves}\leq 2}$ lies in AC^0, while DPPH is L-complete [7].

Organization of the Paper. In Section 2 we formally introduce perfect phylogeny haplotyping problems. Section 3 is devoted to our fixed-parameter algorithm, whose runtime properties are stated in Theorem 3.1. We first develop the techniques needed for its proof in Section 3.1, then we present the actual fixed-parameter algorithm in Section 3.2. Section 4 contains our two hardness results: the NP-completeness of IDPPH with a fixed number of missing entries per SNP site (Theorem 4.1) and the NP-completeness of $\text{IDPP}_{\text{leaves}\leq 2}$ (Theorem 4.2).

2 Preliminaries

Haplotypes, Genotypes, and Perfect Phylogenies. A *haplotype* h is a string over the alphabet $\{0,1\}$, a *genotype* g is a string over the alphabet $\{0,1,2\}$. Two haplotypes h_1 and h_2 of length m *explain* a genotype g of length m, if for every $i \in \{1,\dots,m\}$ we have $g[i] = 2$ if $h_1[i] \neq h_2[i]$ and $g[i] = h_1[i] = h_2[i]$ otherwise. It is customary to formalize sets of genotypes (haplotypes) as matrices in which each row is a genotype (haplotype). The columns represent SNP sites. We say that a haplotype matrix B *explains* a genotype matrix A, if for every row r the genotype in row r of A is explained by the haplotypes in rows $2r-1$ and $2r$ of B.

Haplotyping is the task, given a genotype matrix, to determine an explaining haplotype matrix that correctly predicts the chromosome pairs from which the genotypes have been extracted. Since there may be several explaining haplotype matrices, biological assumptions are used to come up with criteria that narrow down the solution space. Gusfield [15] proposed to use phylogenetic trees for this purpose: A haplotype matrix B *admits a perfect phylogeny* if there exists a tree (an undirected, connected, acyclic graph) T_B such that:

1. Each column of B labels exactly one edge of T_B and each edge is labeled by at least one column.
2. Each row of B labels exactly one node of T_B.
3. For every two rows h_1 and h_2 of B and every column i, we have $h_1[i] \neq h_2[i]$ if, and only if, i lies on the path from h_1 to h_2 in T_B.

$$B = \begin{array}{cccccccccc} & a & b & c & d & e & f & g & h & \\ \left(\right. & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & \left.\right) h_1 \\ & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & h_2 \\ & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & h_3 \\ & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & h_4 \\ & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & h_5 \\ & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & h_6 \\ & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & h_7 \\ & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & h_8 \end{array}$$

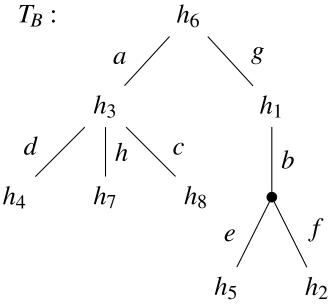

Figure 1: An example haplotype matrix B admitting a perfect phylogeny T_B.

We give an example in Figure 1. The intuition behind the above definition is as follows. The nodes of the tree T_B correspond to haplotypes. The edges between the nodes correspond to mutation events: When we move from one node to another node along a single edge, the label(s) of the edge name exactly those columns in which the node labels differ. This means that when we remove an edge labeled by a column c, the two resulting components have the property that all nodes in one component have a 0 in column c and all nodes in the other component have a 1 in that column.

When a haplotype matrix B admits a perfect phylogeny T_B and, at the same time, explains a genotype matrix A, we also say that A *admits a perfect phylogeny*. In this case, it is useful to define a tree T_A as follows: Its topology is the same as T_B's and so are the node labels, but the edges are labeled by the columns of A (which may contain 2-entries) instead of the columns of B (where each 2-entry is replaced by a 0-entry and a 1-entry). We call T_A *a perfect phylogeny for A*.

Formal Haplotyping Problems. The formal *perfect phylogeny haplotyping* problem (PPH) is the set of all genotype matrices that admit a perfect phylogeny:

Problem PPH
Input A genotype matrix A
Question Does A admit a perfect phylogeny?

If the input matrix contains only haplotypes, no haplotyping needs to be done and the question is just whether they can be arranged in a perfect phylogeny. The resulting problem is known as PP.

If the input genotype matrix A contains at least one genotype that is completely homozygous, then we immediately know one of the sought haplotypes. We can regard the haplotype h as the root of the phylogeny and thereby give the edges an orientation. Since the roles of 0-entries and 1-entries can be exchanged individually for each site, we may assume that h is the all-0-haplotype (just flip 0 and 1 in the columns where h contains 1). Perfect phylogenies containing the all-0-haplotype as a node label are called *directed perfect phylogenies*. The resulting haplotyping problem is called *directed perfect phylogeny haplotyping* (DPPH).

Problem DPPH
Input A genotype matrix A
Question Does A admit a directed perfect phylogeny?

Analogously we define:

Problem DPP
Input A haplotype matrix B
Question Does B admit a directed perfect phylogeny?

To model the imperfectness of the laboratory methods for extracting genotype data we use the symbol "?" to indicate sites where we do not know whether the correct entry is 0, 1, or 2. An *incomplete genotype g* or an *incomplete haplotype h* is a string over the alphabet $\{0,1,2,?\}$ or $\{0,1,?\}$, respectively. A *completion* of g is a string that is obtained by replacing all ?-entries with 0-, 1-, or 2-entries.

Problem IPPH
Input An incomplete genotype matrix A
Question Can A be completed to become a genotype matrix that admits a perfect phylogeny?

The problems IDPPH, IPP, and IDPP are defined analogously. Note that, while PP and DPP are nearly the same problems and can easily be reduced to each other, IPP and IDPP differ more strongly: IPP is NP-complete [23], but IDPP is solvable in polynomial time [2, 19].

Problem Parameters. We parametrize the perfect phylogeny haplotyping problem by two parameters: First, the parameter k denotes the maximum number of ?-entries in any column of the input matrix. Second, the parameter l denotes the maximum number of allowed leaves in an explaining perfect phylogeny for the haplotype matrix.

In order to analyze more easily which influence these parameters have individually and jointly on the complexity of problems like IPPH, we introduce an additional notation: We add the index "leaves$\leq l$" to a problem to indicate that only those input instances are in the language for which an explaining perfect phylogeny having at most l leaves can be found. For instance, the problem $\text{PPH}_{\text{leaves}\leq l}$ can formally be defined as follows:

Problem $\text{PPH}_{\text{leaves}\leq l}$
Input A genotype matrix A
Question Does A admit a perfect phylogeny with at most l leaves?

Note that in some papers, such as [15], the definition of a phylogenetic tree requires all nodes that represent haplotypes to be leaves. Then the number of leaves is exactly the number of different haplotypes in the explaining matrix. In contrast, our definition of a perfect phylogeny, also used in [10], forbids unlabeled edges, and haplotypes may label inner nodes. Here the number of leaves corresponds to the number of different lineages that have developed when interpreting the perfect phylogeny as a pedigree.

Four Gamete Property. For an $n \times m$ matrix S and a matrix A, we say that S is a *submatrix* of A if there are row indices $g_1, \dots, g_n$ and column indices $c_1, \dots, c_m$ such that for all $i \in \{1,\dots,n\}$ and $j \in \{1,\dots,m\}$ it holds that $S[i,j] = A[g_i, c_j]$. Note that a submatrix is not necessarily a connected block in A. The following fact characterizes PP and DPP in terms of submatrices. Part 1 is the well known *four gamete property*, part 2 is its translation to the directed case, which we call the *three gamete property*.

Fact 2.1. *Let B be a haplotype matrix.*

1. *B admits a perfect phylogeny if, and only if, B does not contain the submatrix* $\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$.
2. *B admits a directed perfect phylogeny if, and only if, B does not contain the submatrix* $\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$.

Resolving Heterozygous Sites. The four gamete property has implications for the question of how heterozygous entries (2-entries) of a genotype can be resolved when a perfect phylogeny for the explaining haplotypes is sought. In order to choose two explaining haplotypes h_1 and h_2 for a genotype g

we have to decide for each 2-entry, for which haplotype we put 0 in its place and for which a 1. If for two 2-entries we put the 0's in its place in the same haplotype, we say that the pair is *resolved equally*, otherwise *unequally*. If g and g' are rows in a genotype matrix A that both have a pair of 2-entries in columns c and d, then both pairs must be resolved in the same way (otherwise, the four gamete property would be violated). Thus we always have to resolve the pairs of 2-entries in the columns c and d in the same way and speak about *resolving columns c and d equally* or *unequally*.

The pattern in the following proposition plays an important role in the hardness proofs in Section 4.

Lemma 2.2. *Let A be an incomplete genotype matrix with submatrix* $S = \begin{bmatrix} 1 & 0 & ? \\ 0 & ? & 1 \\ ? & 1 & 0 \end{bmatrix}$ *in columns* c_1*,* c_2*, and* c_3*. Let* A' *be a completion of A that admits a directed perfect phylogeny via an explaining haplotype matrix B. Then*

1. *either the ?-entries in S are replaced by three 0-entries in* A' *and every pair of columns from* $\{c_1, c_2, c_3\}$ *is resolved unequally in B,*
2. *or the ?-entries in S are replaced by two 0-entries and one 1-entry in* A' *and exactly one pair of columns from* $\{c_1, c_2, c_3\}$ *is resolved equally in B.*

Proof. First note that none of the ?-entries can be set to 2, since then we get the submatrix $\begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}$ which does not admit a directed perfect phylogeny because it can only be explained by the forbidden matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$ from Fact 2.1. Also, setting two missing entries to 1 produces the same submatrix. Thus, there are only two possibilities: either filling all ?-entries with 0, or setting one to 1 and the remaining two to 0. In the first case all pairs of columns are resolved unequally. In the second case two column pairs are resolved unequally, the other pair is resolved equally. □

3 Fixed-Parameter Tractability Result

In this section we show that, for every fixed $l \geq 2$, the problem IPPH$_{\text{leaves}\leq l}$ is fixed-parameter tractable with respect to the maximal number of missing entries per column:

Theorem 3.1. *For each* $l \geq 2$*, the problem* IPPH$_{\text{leaves}\leq l}$ *is fixed-parameter tractable with respect to the maximal number of ?-entries per column.*

In the following two sections, we first introduce the new notion of the *light component size* as a generalization of Gusfield's leaf count [15], and an alternative characterization for IPPH. Then we show how this can be used in an algorithm.

3.1 A Characterization of Undirected Perfect Phylogeny Haplotyping

A major tool in the development of efficient algorithms for the DPPH problem has been the *leaf count* of a column, which is twice the number of its 1-entries plus the number of its 2-entries. The name "leaf count" stems from the following observation: In a directed perfect phylogeny for a genotype matrix A, the number of haplotypes below the edge labeled by a column equals exactly its leaf count. This means that if two columns occur on a path from a leaf to the root (recall that this is always the all-0-haplotype in a directed perfect phylogeny), the column with a greater leaf count is located nearer to the root.

For undirected perfect phylogenies the leaf count is no longer meaningful since there is no distinguished root node that is known in advance. To tackle this problem, we introduce the new notion of *light component sizes.* For a column c, let $n_0(c)$, $n_1(c)$, and $n_2(c)$ denote the number of 0-entries, 1-entries, and 2-entries in c, respectively.

Definition 3.2. For a column c of a genotype matrix A its *light component size* and *heavy component size* are defined as follows:

$$\begin{aligned} \operatorname{lcs}(c) &:= n_2(c) + 2 \cdot \min\{n_0(c), n_1(c)\}, \\ \operatorname{hcs}(c) &:= n_2(c) + 2 \cdot \max\{n_0(c), n_1(c)\}. \end{aligned}$$

$$A = \begin{array}{c} \begin{array}{cccccccc} a & b & c & d & e & f & g & h \end{array} \\ \begin{pmatrix} 1 & 2 & 0 & 0 & 1 & 2 & 1 & 1 \\ 0 & 1 & 0 & 2 & 1 & 1 & 0 & 1 \\ 1 & 2 & 0 & 0 & 2 & 1 & 2 & 1 \\ 0 & 1 & 2 & 0 & 1 & 1 & 0 & 2 \end{pmatrix} \end{array}$$

column x	a	b	c	d	e	f	g	h
$\operatorname{lcs}(x)$	4	2	1	1	1	1	3	1
$\operatorname{lcv}(x)$	1	0	1	1	0	0	1	0

$b \preceq e, f$

$b \succeq a, g$

$b \perp c, d, h$

Figure 2: Example of a genotype matrix A with the light component sizes and values of its columns. For column b we show how it relates to the other columns. The matrix A can be explained by the haplotype matrix B from Figure 1 and therefore admits the perfect phylogeny T_B from Figure 1.

The key observation is that when we remove an edge labeled by a column c from a perfect phylogeny T_A, then two components result and the number of node labels in one of these components will be $\operatorname{lcs}(c)$ and we call the component the *light component*, the other will contain $\operatorname{hcs}(c)$ labels and we call it the *heavy component*. (In case $\operatorname{lcs}(c) = \operatorname{hcs}(c)$, the choice is arbitrary.) To see this, recall that in one component all node labels have a 0 in column c (and a 1 in the other component). Each of the $n_0(c)$ many 0-entries of c contributes two node labels to this component, while each 2-entry contributes one node label, which means that the number of node labels in this component is either $\operatorname{lcs}(c)$ or $\operatorname{hcs}(c)$. The argument is similar for the other component and for 1-entries.

We have just seen that the value in column c of all node labels of the light component is the same. Let us call this value the *light component value* $\operatorname{lcv}(c)$. Clearly, $\operatorname{lcv}(c) = 0$ if $n_0(c) < n_1(c)$ and $\operatorname{lcv}(c) = 1$ if $n_0(c) > n_1(c)$. For $n_0(c) = n_1(c)$ we remarked earlier that the light component can be chosen arbitrarily; at this point we implicitly fix that choice by setting $\operatorname{lcv}(c) = 1$. Symmetrically, the value in column c of the node labels of the heavy component are all the same and equal to $\operatorname{hcv}(c) = 1 - \operatorname{lcv}(c)$. See Figure 2 for an explaining example.

Our next aim is to define a quasi-ordering $\preceq$ on columns that helps us to arrange them in perfect phylogenies. Suppose that for two columns c and d we know that the light component of d is a superset of the light component of c. Consider a node label l and suppose the value of l at the position of column c happens to be the light component value of c. Then we know that l must lie in the light component of c and, thus, also in the light component of d, which in turn means that at position d in l we must have the light component value of d. Phrased more succinctly: for every $i \in \{1, \dots, n\}$ we have $c[i] = \operatorname{lcv}(c) \implies d[i] = \operatorname{lcv}(d)$ and, by a similar argument, also $d[i] = \operatorname{hcv}(d) \implies c[i] = \operatorname{hcv}(c)$. Let us write $c \preceq d$ whenever these two implications hold for every i. Then $c \preceq d$ is a necessary, but not a sufficient, condition for c's light component being contained in d's light component. We remark that $c \preceq d$ implies $\operatorname{lcs}(c) \leq \operatorname{lcs}(d)$. We can similarly consider the case of columns whose light components are disjoint: then for every $i \in \{1, \dots, n\}$ we have $c[i] = \operatorname{lcv}(c) \implies d[i] = \operatorname{hcv}(d)$ and $d[i] = \operatorname{lcv}(d) \implies c[i] = \operatorname{hcv}(c)$. We write $c \perp d$ whenever these two implications hold for every i. The introduced notions are related to Gusfield's leaf count. Consider the haplotype that, at each site, has the column's corresponding heavy component value. The proof of Lemma 3.4 shows that adding this haplotype as a root r to a perfect phylogeny and rearranging some columns locally, implies the following properties: (1) If we consider the columns on a path c_1, c_2, $\dots$, c_k from r to a leaf, then on this path, the columns with greater light component size are located nearer to the root and $c_1 \succeq c_2 \succeq \cdots \succeq c_k$. (2) If two columns c and d label edges that are incident to a common node v and do not lie on the path from r to v, then $c \perp d$. (3) Moreover, the 2-entries of each genotype that is explained by the perfect phylogeny still make up a path in it.

Our algorithm is only concerned with building a tree whose edges are labeled with columns of the input genotype matrix; the nodes of the tree are not labeled and the rows of the explaining haplotype matrix B are irrelevant to the algorithm. Since edge labels correspond to mutation events, we call the tree constructed by the algorithm a *mutation tree*.

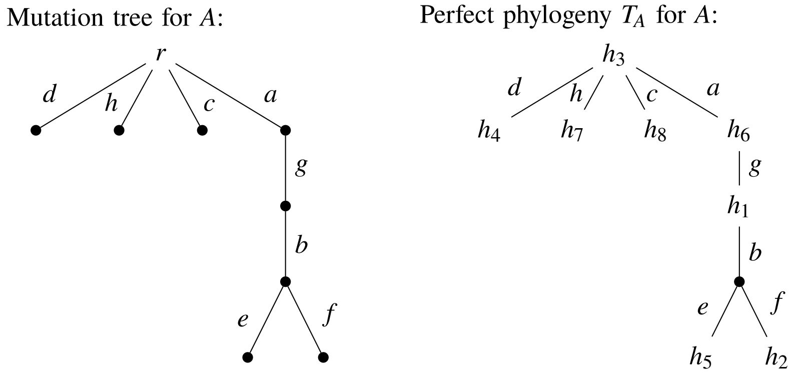

Figure 3: The mutation tree for genotype matrix A from Figure 2 and the obtained perfect phylogeny explaining A. One can see that T_A equals T_B from Figure 1.

Definition 3.3. Let A be a genotype matrix. A *mutation tree T for A* is an undirected tree whose edges are bijectively labeled by A's columns and which has a distinguished root node r such that:

1. *Ordering condition:* For every path originating at the root with edge labels c_1, c_2, ..., c_k we have $c_1 \succeq c_2 \succeq \cdots \succeq c_k$.
2. *Compatibility condition:* For every two columns c and d that are incident to a common node v and that do not lie on the path from r to v we have $c \perp d$.
3. *Two-path condition:* For every three columns c, d, and e that are incident to the same node, there is no $i \in \{1,\dots,n\}$ such that $c[i] = d[i] = e[i] = 2$.

For an example we refer to Figure 3.

Lemma 3.4. *A genotype matrix A admits a perfect phylogeny with l leaves if, and only if, there exists a mutation tree for A with l leaves.*

Proof. For the only-if-direction let A be a genotype matrix, T_A a perfect phylogeny for A, and B an explaining haplotype matrix. We may assume that each edge of T_A is labeled exactly once, otherwise replace edges with multiple labels by paths of appropriate lengths in which each edge has a unique label. We argue that T_A with an appropriate root and without node labels (and, in some cases, some minor additional changes) is a mutation tree for A.

Let $\{d_1,\dots,d_s\}$ be the set of all columns with maximal light component size. We claim that this set forms a connected component in T_A and $\mathrm{lcs}(d_1) = \cdots = \mathrm{lcs}(d_s)$. For the proof, we distinguish two cases. First, if $\mathrm{lcs}(d_1) = \cdots = \mathrm{lcs}(d_s) < \mathrm{hcs}(d_1) = \cdots = \mathrm{hcs}(d_s)$, there are no two columns d_i and d_j such that d_i belongs to the light component of d_j and vice versa (otherwise $\mathrm{lcs}(d_i) \geq \mathrm{hcs}(d_j) = \mathrm{hcs}(d_i)$, which contradicts $\mathrm{lcs}(d_i) < \mathrm{hcs}(d_i)$). This assures the existence of a column $c \in \{d_1,\dots,d_s\}$ that lies in the heavy component of every other column d_i. Second, if $\mathrm{lcs}(d_1) = \cdots = \mathrm{lcs}(d_s) = \mathrm{hcs}(d_1) = \cdots = \mathrm{hcs}(d_s)$, assume there are d_i, d_j, and d_k with maximal light component size such that all three of them are incident to a common node v. Then the component of d_i that contains v also contains the components of d_j and d_k that do not contain v. So $\operatorname{lcs}(d_i) \geq \operatorname{lcs}(d_j) + \operatorname{lcs}(d_k)$, which contradicts $\operatorname{lcs}(d_i) = \operatorname{lcs}(d_j) = \operatorname{lcs}(d_k)$. Thus, the d_i's form a path and no haplotypes label its inner nodes. Hence, the columns on this path can be rearranged so that one of the columns, call it c, lies in the heavy component of every other column. In both cases let r be the node incident to c in its heavy component. Then r lies in the heavy component of each column d_i. For columns d with $\operatorname{lcs}(d) < \operatorname{lcs}(c)$ the light component of d contains neither the light component nor the heavy component of c. Thus, regardless of whether d lies in the light component or heavy component of c, the node r lies in the heavy component of d.

We show that T_A with the edges d_i possibly rearranged as described earlier, and with root r is a mutation tree when we disregard the labeling of its nodes. To show that the ordering condition is satisfied, consider two columns c_1 and c_2 where c_1 lies on the path from r to c_2. Since r lies in the heavy component of each column, the heavy component of c_1 is a subtree of the heavy component of c_2 and the light component of c_2 is a subtree of the light component of c_1, which implies $c_1 \succeq c_2$. For the compatibility condition consider two edges c and d that are incident to a node v and that do not lie on a path from r to v. Then the light components of c and d are disjoint and, thus, $c \perp d$. Finally, we verify the two-path condition. Consider a genotype in A with a 2-entry in column d. One of the explaining haplotypes in B contains 0 and the other one contains 1 in column d, so in T_A one of them labels a node in the light component and the other labels a node in the heavy component of d. If we have three columns c_1, c_2, and c_3 incident to a common node v, then the components of the columns that do not include v are disjoint. Hence, if A contains a genotype g with 2-entries in these columns, we have three disjoint components which must each contain a label for one of the two explaining haplotypes for g. Since this is a contradiction, the 2-path condition is true.

To prove the if-direction, let T be a mutation tree with root r. We show that the following node labeling makes T a perfect phylogeny for A: For each column c and node v, we set the label of v in column c to $\operatorname{lcv}(c)$ whenever c lies on the path between r and v and to $\operatorname{hcv}(c)$, otherwise. It suffices to show that for every genotype $g \in A$ there are two labels explaining it. Let g be a genotype from A and let $A_g^{\mathrm{lcv}} := \{c \mid c \text{ is column of } A,\ g[c] = \operatorname{lcv}(c)\}$ and $A_g^2 := \{c \mid c \text{ is column of } A,\ g[c] = 2\}$. In the following we show that (a) there is a node v that is connected to r via a path labeled exactly by the columns in A_g^{lcv}, (b) there are two nodes w and w' connected by a path that goes through v and is labeled exactly by the columns in A_g^2, and (c) the labels of w and w' explain g.

Since T satisfies the ordering condition, the columns from A_g^{lcv} form a connected component in T that contains r. The compatibility condition ensures that this component is a single path from r to some node v. Thus, (a) is true. Let T_v be the subtree of T rooted at v. Assume there is a $c \in A_g^2$ that labels an edge not belonging to T_v. Let v' be the least common ancestor of c and v. Let d_1 and d_2 be the columns connected to v' such that d_1 starts the path from v' to v and d_2 starts the path from v' to c. Then, due to the compatibility condition, it holds $d_1 \perp d_2$ and therefore $g[d_2] = \operatorname{hcv}(d_2)$. The ordering condition gives $d_2 \succeq c$, which implies $g[c] = \operatorname{hcv}(c)$, a contradiction to the choice of c. So, all columns in A_g^2 are in T_v. With the property that all columns from T_v have a heavy component value or a 2-entry in g, the columns in A_g^2 form a connected component in T_v that contains v. The two-path condition implies that this component is a path, so (b) holds. Finally, to prove (c) let w and w' be the two vertices that are connected by this A_g^2-path. The path from r to v contains exactly the columns with light component value in g, so the labels both have a light component value in these columns. The columns from $c \in A_g^2$ are distributed among the paths from v to w and v to w'. Therefore one label has $\operatorname{lcv}(c)$ and the other $\operatorname{hcv}(c)$ in column c. All columns not belonging to A_g^{lcv} or A_g^2 do not appear in either path, thus the labels contain heavy component values in these columns. Hence, the labels of w and w' explain g. □

3.2 The Fixed-Parameter Algorithm

Our fixed-parameter algorithm for $\text{IPPH}_{\text{leaves}\le l}$ works in two stages. The first stage is a preprocessing of the input matrix. After the preprocessing, the maximal number of columns with the same light component size is bounded by a function in l and k, where k is the maximum number of missing entries per column. The basic idea is that if there are many different columns with the same light component size, they must lie on many different branches and, thus, at some point it is no longer possible to arrange them in a perfect phylogeny with only l leaves. The following lemma states the effect of the preprocessing precisely.

Lemma 3.5. *There is an algorithm* PREPROCESS *that gets an incomplete $n \times m$ genotype matrix A with at most k many ?-entries per column as input and outputs, in time $O(k4^k m^3 n)$, a genotype matrix A' such that:*

1. *$A \in \text{IPPH}_{\text{leaves}\le l}$ if, and only if, $A' \in \text{IPPH}_{\text{leaves}\le l}$.*
2. *There are no duplicate columns in A' and no columns that can be completed to a constant column.*
3. *For every i there are at most $(2k+1)l(3l)^k k!$ columns in A' with light component size i.*

Proof. We just give a brief sketch because both, algorithm and proof, are straightforward generalizations of those given by Gramm et al. [13] for the case $l = 2$. The main idea we exploit is as follows: if the input matrix contains many columns whose missing entries can be filled such that all obtained columns are equal, we make no mistake in filling them that way. More precisely, one proves the following:

Let A be a genotype matrix with at most k missing entries per column. Let $h \ge 0$ be the minimal number such that there is a subset C of columns from A with $|C| > (3l)^h h$ and a (possibly new) "consensus genotype" c with exactly $k-h$ missing entries that can be obtained from every genotype in C by filling some of its missing entries. Let A' be the incomplete genotype matrix obtained from A by deleting all columns from C and adding c. Then $A \in \text{IPPH}_{\text{leaves}\le l}$ if and only if $A' \in \text{IPPH}_{\text{leaves}\le l}$.

In the preprocessing phase columns are replaced according to this rule to obtain a genotype matrix that satisfies the properties stated in the lemma. □

The second stage is the main part of the algorithm. By Lemma 3.4, in order to decide whether the preprocessed version A has the property $A \in \text{IPPH}_{\text{leaves}\le l}$, it suffices to test whether A can be completed in such a way that it admits a mutation tree with at most l leaves.

For the presentation of our algorithm we need some additional terminology: Given a set of columns A or a matrix A, let $A|_{\text{lcs}=i}$, $A|_{\text{lcs}\le i}$, and $A|_{\text{lcs}>i}$ denote the set of all columns c of A with $\text{lcs}(c) = i$, $\text{lcs}(c) \le i$, and $\text{lcs}(c) > i$, respectively. A *completion* of a set of columns with ?-entries is obtained by replacing all ?-entries by 0-, 1-, or 2-entries. Note that a completion of a column with light component size i can have a light component size between i and $i+2k$, where k is, as always, the number of ?-entries in the column. The *inner part* of a mutation tree T is the set $\text{inner}(T)$ of edges that are incident to the root of T.

The mutation tree construction algorithm works in iterations $i = 1, 2, \dots, n$. In iteration i it processes all completions of the set $A|_{\text{lcs}=i}$. The algorithm keeps track of what it has already found out about completions of $A|_{\text{lcs}<i}$ in previous iterations in what we call *tree records* (I, λ, U). Such a record consists of an *inner part* I, a number $\lambda \in \{0, \dots, l\}$ of leaves, and a set of *unprocessed columns* U. The following definition formalizes the properties that tree records should have:

Definition 3.6. Let A be an incomplete $n \times m$ genotype matrix and let $i \in \{1, \dots, n\}$. A tree record (I, λ, U) is *good for A and i* if there exists a completion S_i of $A|_{\text{lcs}\le i}$ such that (a) $I = \text{inner}(T_i)$ for some mutation tree T_i for $S_i|_{\text{lcs}\le i}$, (b) λ is the number of leaves of T_i, and (c) $U = S_i|_{\text{lcs}>i}$.

The job of the algorithm is to compute in each iteration i the set R_i of all good tree records for A and i. Clearly, if R_n is nonempty after the last iteration, there exist a completion for A and a mutation tree for it with at most l leaves; and otherwise no such completion exists. Figure 4 shows the pseudo-code of the algorithm, Figure 5 shows an example of the algorithm in action.

```
Algorithm SOLVE-IPPH$_{\text{leaves}\leq l}$.
Input: An $n \times m$ genotype matrix $A$ with at most $k$ missing entries per column.
1  $A \leftarrow$ PREPROCESS$(A)$
2  $R_0 \leftarrow \{(\emptyset, 0, \emptyset)\}$
3  for increasing light component sizes $i \leftarrow 1, 2, \dots, n$ do
4      $R_i \leftarrow \emptyset$
5      for each completion $C$ of $A|_{\text{lcs}=i}$ do
6          for each tree record $(I, \lambda, U) \in R_{i-1}$ do
7              for each mutation tree $T$ for $I \cup C|_{\text{lcs}=i} \cup U|_{\text{lcs}=i}$
                          with $\lambda' - \lambda + |I|$ leaves for some $\lambda' \leq l$
                          where all columns from $I$ are incident to leaves of $T$ do
8                  $R_i \leftarrow R_i \cup \{(\text{inner}(T), \lambda', C|_{\text{lcs}>i} \cup U|_{\text{lcs}>i})\}$
9  if $R_n$ is nonempty then output "$A \in$ IPPH$_{\text{leaves}\leq l}$" else output "$A \notin$ IPPH$_{\text{leaves}\leq l}$"
```

Figure 4: Our decision algorithm for IPPH$_{\text{leaves}\leq l}$.

The following two lemmas imply that the algorithm is correct and that it is a fixed-parameter algorithm for IPPH$_{\text{leaves}\leq l}$. Together, they prove Theorem 3.1.

Lemma 3.7. *After each iteration i of algorithm* SOLVE-IPPH$_{\text{leaves}\leq l}$*, the set R_i contains exactly the good tree records for A and i.*

Proof. We prove the claim by induction over i. For $i = 0$ the initialization $R_0 = \{(\emptyset, 0, \emptyset)\}$ is correct because the preprocessing ensures that $A|_{\text{lcs}=0} = \emptyset$, see the second property of Lemma 3.5.

For the inductive step from $i-1$ to i, we first argue that the algorithm adds only good tree records to R_i. Suppose the algorithm adds some tree record (I', λ', U') to R_i. Then there exists a completion C of $A|_{\text{lcs}=i}$ and some tree record $(I, \lambda, U) \in R_{i-1}$, such that there is a mutation tree T passing the test from line 7. By the inductive hypothesis, (I, λ, U) is good as witnessed by some T_{i-1} and S_{i-1}. Our objective is to combine T_{i-1} and T into a a mutation tree T_i for all of $S_i|_{\text{lcs}\leq i} = S_{i-1}|_{\text{lcs}\leq i-1} \cup C|_{\text{lcs}=i} \cup U|_{\text{lcs}=i}$. First, we split T_{i-1} at the root, which leads to $|\text{inner}(T_{i-1})|$ many subtrees. We then identify the top edges of these subtrees with the edges that are labeled by columns from $\text{inner}(T_{i-1})$ in T. Remember that these columns label edges that are incident to leaves. Figure 6 shows the construction of T_i from T and T_{i-1}. The tree T_i is a mutation tree since the local properties at incident edges and edges around nodes are satisfied by construction. Since T has $\lambda' - \lambda + |I| \leq l - \lambda + |I|$ leaves and T_{i-1} has λ leaves, T_i has $\lambda' \leq l$ leaves.

It remains to argue that all good tree records (I', λ', U') are added to R_i. Let S_i and T_i witness that the tree record is good for A and i. Partition S_i into two sets S_{i-1} and C, such that S_{i-1} is a completion of $A|_{\text{lcs}\leq i-1}$ and C is a completion of $A|_{\text{lcs}=i}$. Let T_{i-1} be obtained from T_i by contracting all edges labeled by columns with light component size i.

We claim that T_{i-1} is a mutation tree for the columns from $S_{i-1}|_{\text{lcs}\leq i-1}$. To prove this, we show that for every mutation tree T' and every edge $\{v, w\}$ labeled d, where v lies on the path from the root to w, if we contract the edge $\{v, w\}$, the resulting tree T'' still satisfies the ordering, compatibility, and two-path conditions:

1. T'' clearly still satisfies the ordering condition.

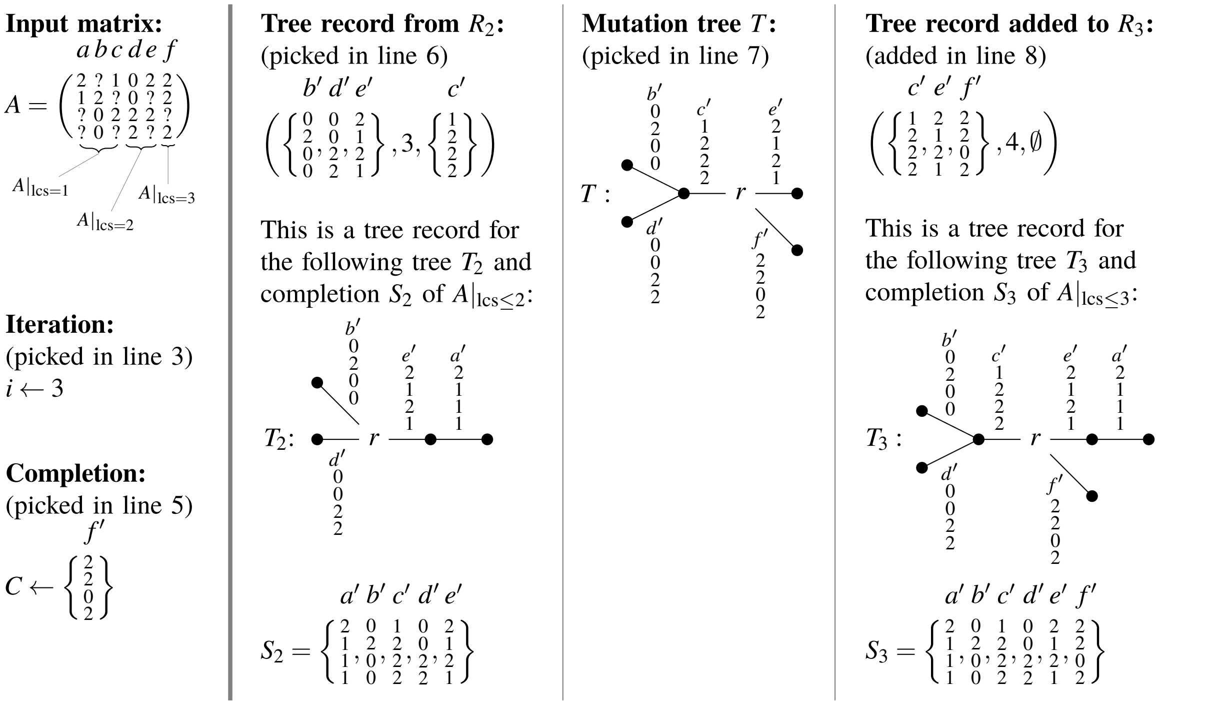

Figure 5: Example of the third iteration of SOLVE-IPPH$_{\text{leaves}\leq l}$ for the indicated input matrix A. We depict a set of possible values for the loop variables for which a new tree record is added to R_3.

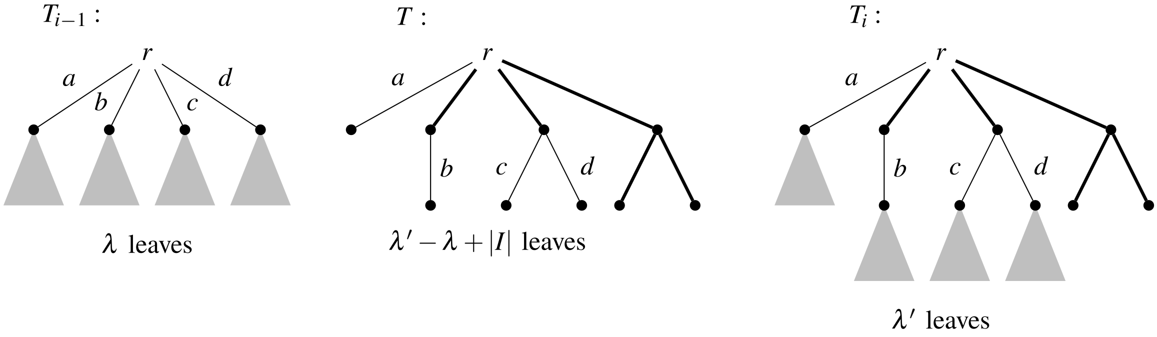

Figure 6: Example of the construction of T_i from T and T_{i-1}. In line 7, the algorithm combines edges that are labeled with columns inner(T_{i-1}) (thin lines) and edges that are labeled with columns in $C|_{\text{lcs}\leq i} \cup U|_{\text{lcs}\leq i}$ (thick lines) to construct a mutation tree T for them, such that the columns from inner(T_{i-1}) label edges that are incident to leaves. Since the ordering, compatibility and two-path conditions are satisfied locally in trees T_{i-1} and T, there exists the depicted combined mutation tree T_i for $S_{i-1}|_{\text{lcs}\leq i-1} \cup C|_{\text{lcs}=i} \cup U|_{\text{lcs}=i}$. We also know that T_i has $\lambda' \leq l$ leaves since T_{i-1} has λ leaves and T has $\lambda' - \lambda + |I| \leq l - \lambda + |I|$ leaves.

2. For the compatibility condition, let u be a node of T''. If u is neither v nor w, the compatibility condition is still true at u. Otherwise, $u = v = w$. Suppose for sake of contradiction, that u is incident to two columns c and c' that are not on the path to the root and where $c \perp c'$ does not hold. Without loss of generality, we may assume $c[g] = \mathrm{lcv}(c)$ and $c'[g] \neq \mathrm{hcv}(c')$ for a genotype g. Since the compatibility condition holds for T', we know that neither v nor w is incident to both c and c' in T'. First, consider the case that v is incident to c and w is incident to c'. Since $c[g] = \mathrm{lcv}(c)$, we have $d[g] = \mathrm{hcv}(d)$. Together with the ordering condition it follows that c' must have a heavy component value in genotype g, a contradiction. Second, assume that v is incident to c' and w is incident to c. From the ordering condition we can deduce that d has a light component value in genotype g. Thus, c' and d at node v in T' contradict the compatibility condition.
3. For the two-path condition, let u be a node of T''. Again, if u is neither v nor w, the two-path condition is still satisfied at u, so assume $u = v = w$. For the sake of contradiction, assume that u is incident to three columns c, c', and c'' with $c[g] = c'[g] = c''[g] = 2$ for a genotype g. Since the two-path condition holds in T', the nodes v and also w are incident to at least one of these columns. First, we assume that v is incident to c and w is incident to c' and c''. The ordering condition implies that d has either a 2-entry or a light component value in g. If d has a 2-entry in g, the two-path condition is not satisfied at node w. If d has a light component value in g, we distinguish the cases that c labels the path between v and the root in T' and that c does not label this path. In the first case, the ordering condition is not satisfied for c and d, and in the second case the compatibility condition is not satisfied at node v. If we assume that two columns c and c' are incident to v and one column c'' is incident to w, we can analogously deduce a contradiction. Thus, the two-path condition holds for every node of T''.

We have now proved that T_{i-1} is a mutation tree for $S_{i-1}|_{\mathrm{lcs} \leq i-1}$. By the inductive assumption, there must be good tree record $(\mathrm{inner}(T_{i-1}), \lambda, S_{i-1}|_{\mathrm{lcs}>i-1}) \in R_{i-1}$ for T_{i-1} and S_{i-1}. Since light component sizes only increase as we near the root, in T_i there can be no column c with $\mathrm{lcs}(c) \leq i-1$ that labels an edge between the root and a column c' with $\mathrm{lcs}(c') = i$. With this property in mind, we combine $\mathrm{inner}(T_{i-1})$ and the columns with light component size exactly i from S_i to a tree T with $\mathrm{inner}(T) = \mathrm{inner}(T_i)$ that has $\lambda' - \lambda + |\mathrm{inner}(T_{i-1})| \leq l - \lambda + |\mathrm{inner}(T_{i-1})|$ leaves. This construction is the converse of the construction in Figure 6. Then T will pass the test of line 7 and $(\mathrm{inner}(T_i), \lambda', U')$ will, indeed, be inserted into R_i. □

Lemma 3.8. *Algorithm* SOLVE-IPPH$_{\mathrm{leaves} \leq l}$ *runs in time* $O\big(f(k)m^l n^2\big)$, *where* f *is an at most double exponential function depending only on* k.

Proof. By Lemma 3.5 the preprocessing takes time $O(k4^k m^3 n)$. The number of completions of $A|_{\mathrm{lcs}=i}$ considered in round i is bounded by $3^{k(2k+1)l(3l)^k k!}$ since Lemma 3.5 limits the size of $A|_{\mathrm{lcs}=i}$ by $(2k+1)l(3l)^k k!$. We now argue that the number of good tree records over which the algorithm iterates can be at most $3^{k|A|_{i-2k \leq \mathrm{lcs} \leq i-1}|}$. A tree record consists of a set I of at most l complete columns for which there are at most $m^l 3^{kl}$ possibilities, a value $\lambda \leq l$, and a set U of unprocessed completions. The set U contains complete columns from the preceding iterations that have light component size at least i. When we complete a column, its light component size can only increase by at most $2k$ and, thus, a complete column corresponds to an incomplete column from the last $2k$ rounds. Thus there are at most $3^{k|A|_{i-2k \leq \mathrm{lcs} \leq i-1}|}$ possibilities, which is also bounded by a function in k and l. Finally, for the runtime of the inner loop, just note that the size of $I \cup C|_{\mathrm{lcs}=i} \cup U|_{\mathrm{lcs}=i}$ depends only on l and k. □

4 Hardness Results

In this section we show that both parameters, the number of ?-entries per column and the number of leaves in the phylogeny, cannot be used individually, but only in tandem to arrive at an efficient IPPH-algorithm. We show that IPPH is neither fixed-parameter tractable when parametrized by the number of leaves, nor when parametrized by the number of missing entries per column, unless $\mathrm{P} = \mathrm{NP}$. In fact, we show that IPPH stays NP-hard if we fix one of the parameters to a constant and is, thus, not even in XP unless $\mathrm{P} = \mathrm{NP}$.

In the proofs we use the following NP-complete [22] variants of the satisfiability problem to reduce from.

Problem MONOTONE 1IN3SAT 3OCC
Input A propositional formula $\phi = \bigwedge_{i=1}^{m} C_i$, where each clause $C_i = (x_0^i \vee x_1^i \vee x_2^i)$ consists of three variables and every variable occurs at most three times
Question Is there a satisfying truth assignment for the variables, mapping exactly one variable per clause to 1?

Problem MONOTONE NAE3SAT
Input A propositional formula $\phi = \bigwedge_{i=1}^{m} C_i$, where each clause $C_i = (x_0^i \vee x_1^i \vee x_2^i)$ consists of three variables
Question Is there a satisfying truth assignment for the variables, such that not all variables in a clause share the same value?

Our first hardness result improves upon a result by Kimmel and Shamir, who have shown in [17] that IDPPH is NP-complete. Our aim is to show that the completeness still holds when the number of ?-entries per row and per column is restricted by a constant, which implies that IDPPH parametrized by the number of missing entries per column is not in XP (unless $\mathrm{P} = \mathrm{NP}$). In order to analyze the number of ?-entries in the genotype matrices generated during the reduction, we need to abandon the graph-theoretic methods used by Kimmel and Shamir and, instead, use a more direct approach based on specifically constructed submatrices.

Theorem 4.1. IDPPH *for instances with up to* 15 *missing entries per column and up to* 5 *missing entries per row is* NP*-complete.*

Proof. Membership in NP is clear. For the NP-hardness we reduce from MONOTONE 1IN3SAT 3OCC. Let $\phi = \bigwedge_{i=1}^{m} C_i$ be an instance of MONOTONE 1IN3SAT 3OCC with m clauses $C_i = (x_0^i \vee x_1^i \vee x_2^i)$ and variables from $V = \{v_1, \dots, v_n\}$. We construct a genotype matrix A with $2n + 12m$ columns and $31m$ rows. The columns are named as follows:

- For each variable $v \in V$ there are two columns v and v'.
- For each clause C_i there are columns $c_j^i, d_j^i, e_j^i, f_j^i$ with $j \in \{0, 1, 2\}$.

We now define the rows of A. For each clause C_i there is a row r^i with 2-entries in columns c_0^i, c_1^i, c_2^i and 0-entries in all other columns. Additionally, for every clause $C_i = (x_0^i \vee x_1^i \vee x_2^i)$ and every $j \in \{0, 1, 2\}$ there are ten rows containing the following submatrix A_j^i and being 0 in all other columns:

$$A^i_j = \begin{array}{cccccccc} x^i_j & {x^i_j}' & d^i_j & e^i_j & f^i_j & c^i_j & c^i_{j'} & c^i_{j''} \\ \hline 2 & 2 & 2 & 2 & 0 & ? & 0 & ? \\ ? & ? & 2 & 2 & 2 & ? & ? & ? \\ ? & 0 & 2 & 0 & 2 & 2 & 2 & ? \\ ? & 0 & 1 & 0 & 0 & ? & 0 & ? \\ ? & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & ? & 1 & 0 \\ 0 & ? & ? & 1 & 0 & ? & 0 & ? \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ ? & 0 & ? & 0 & 1 & 0 & ? & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & ? \end{array}$$

Here $j' = (j+1) \bmod 3$ and $j'' = (j+2) \bmod 3$. From now on we interpret all lower indices in this proof modulo 3.

The idea of this construction is as follows: depending on how we resolve the 2-entries in columns x^i_j and ${x^i_j}'$, there is only one possibility (depicted in Figure 7) to fill the missing entries (except those in column c^i_{j+2}) and resolve the 2-entries in the rest of a submatrix A^i_j without producing the submatrix $\left[\begin{smallmatrix}1&0\\0&1\\1&1\end{smallmatrix}\right]$.

Resolving x^i_j and ${x^i_j}'$ equally or unequally will correspond to setting x^i_j to 0 or 1, respectively. In Figure 7 one can see that c^i_j and c^i_{j+1} are resolved equally if and only if x^i_j and ${x^i_j}'$ are resolved unequally.

For the correctness of the reduction we first show that if ϕ has a solution, then A can be completed to admit a directed perfect phylogeny. Let $\tau : V \to \{0,1\}$ be an assignment that maps exactly one variable of each clause in ϕ to 1. In the submatrices A^i_j with corresponding clause $C_i = (x^i_0 \vee x^i_1 \vee x^i_2)$ we resolve the 2-entries in columns x^i_j and ${x^i_j}'$ equally if $\tau(x^i_j) = 0$, and unequally if $\tau(x^i_j) = 1$. Then we fill and resolve the matrix according to Figure 7. Without loss of generality we assume $\tau(x^i_0) = 1$ and $\tau(x^i_1) = \tau(x^i_2) = 0$. In c^i_0 we fill the remaining ?-entries by copying the entries from c^i_1. The remaining ?-entries of c^i_1 and c^i_2 we set to 0. Finally we resolve the 2-entries in row r^i by resolving (c^i_0, c^i_1) equally and (c^i_1, c^i_2) and (c^i_2, c^i_0) unequally. The resulting haplotype matrix admits a perfect phylogeny with the all-0-haplotype as root.

Now we prove that, if A admits a directed perfect phylogeny, then ϕ has a solution. Let B be the haplotype matrix that explains a genotype matrix A', which can be obtained from A by filling its missing entries. Let $\tau : V \to \{0,1\}$ be an assignment with $\tau(v) = 0$ if columns v and v' are resolved equally and $\tau(v) = 1$ otherwise. We show that τ maps exactly one variable per clause to 1. We consider the clause $C_i = (x^i_0 \vee x^i_1 \vee x^i_2)$. The columns c^i_0, c^i_1, c^i_2 contain the following submatrix in A:

$$\begin{array}{ccc} c^i_0 & c^i_1 & c^i_2 \\ \hline 2 & 2 & 2 \\ 1 & 0 & ? \\ 0 & ? & 1 \\ ? & 1 & 0 \end{array}$$

The first row is r^i, the others are the last rows of A^i_0, A^i_1, and A^i_2. Due to Lemma 2.2 the last three rows ensure that at least two of the three possible column pairs (c^i_0, c^i_1), (c^i_1, c^i_2), and (c^i_2, c^i_0) must be resolved unequally. Because of the first row, it is not possible to resolve all three pairs unequally, therefore exactly one pair must be resolved equally. Thus, τ maps exactly one variable per clause to 1.

x^i_j	$x^{i\,\prime}_j$	d^i_j	e^i_j	f^i_j	c^i_j	$c^i_{j'}$	$c^i_{j''}$
$\substack{0\\1}$	$\substack{1\\0}$	$\substack{0\\1}$	$\substack{1\\0}$	0	$\textcircled{0}$	0	$\textcircled{?}$
$\textcircled{\substack{0\\1}}$	$\textcircled{\substack{1\\0}}$	$\substack{0\\1}$	$\substack{1\\0}$	$\substack{0\\1}$	$\textcircled{0}$	$\textcircled{0}$	$\textcircled{?}$
$\textcircled{\substack{0\\1}}$	0	$\substack{0\\1}$	0	$\substack{0\\1}$	$\substack{1\\0}$	$\substack{1\\0}$	$\textcircled{?}$
$\textcircled{1}$	0	1	0	0	$\textcircled{0}$	0	$\textcircled{?}$
$\textcircled{0}$	1	0	0	0	0	0	0
0	0	0	0	0	$\textcircled{1}$	1	0
0	$\textcircled{1}$	$\textcircled{0}$	1	0	$\textcircled{0}$	0	$\textcircled{?}$
1	0	0	0	0	0	0	0
$\textcircled{1}$	0	$\textcircled{1}$	0	1	0	$\textcircled{0}$	0
0	0	0	0	0	1	0	$\textcircled{?}$

x^i_j	$x^{i\,\prime}_j$	d^i_j	e^i_j	f^i_j	c^i_j	$c^i_{j'}$	$c^i_{j''}$
$\substack{0\\1}$	$\substack{0\\1}$	$\substack{1\\0}$	$\substack{1\\0}$	0	$\textcircled{\substack{1\\0}}$	0	$\textcircled{?}$
$\textcircled{0}$	$\textcircled{0}$	$\substack{0\\1}$	$\substack{0\\1}$	$\substack{1\\0}$	$\textcircled{\substack{0\\1}}$	$\textcircled{\substack{1\\0}}$	$\textcircled{?}$
$\textcircled{0}$	0	$\substack{0\\1}$	0	$\substack{1\\0}$	$\substack{0\\1}$	$\substack{1\\0}$	$\textcircled{?}$
$\textcircled{0}$	0	1	0	0	$\textcircled{1}$	0	$\textcircled{?}$
$\textcircled{1}$	1	0	0	0	0	0	0
0	0	0	0	0	$\textcircled{0}$	1	0
0	$\textcircled{0}$	$\textcircled{1}$	1	0	$\textcircled{1}$	0	$\textcircled{?}$
1	0	0	0	0	0	0	0
$\textcircled{0}$	0	$\textcircled{0}$	0	1	0	$\textcircled{1}$	0
0	0	0	0	0	1	0	$\textcircled{?}$

Figure 7: The submatrix A^i_j filled and resolved when $(x^i_j, x^{i\,\prime}_j)$ are resolved unequally (left) and equally (right). Positions that are ?-entries in A^i_j are indicated by circles.

Finally we count the number of ?-entries. It is easy to see that there are at most five ?-entries in each row. For the columns we first consider c^i_j for $i \in \{0,\dots,m-1\}$, $j \in \{0,1,2\}$. This column has four ?-entries in submatrix A^i_j, six in A^i_{j+1}, and two in A^i_{j+2}, which makes a total of 12 ?-entries. For a variable $v \in V$ the columns v and v' appear in a submatrix A^i_j if and only if v is the jth variable in clause C_i, therefore we have five ?-entries in v and three in v' for every occurrence of v in ϕ. Since every variable occurs at most three times, we get at most 15 ?-entries in column v and at most nine in v'. All columns d^i_j, e^i_j, f^i_j contain only up to two ?-entries. Hence, in each column there are at most 15 missing entries. □

Our second hardness result is the NP-completeness of $\text{IDPP}_{\text{leaves}\leq l}$. This problem reduces to many other problems. First, it is easy to see (but not trivial) that $\text{IDPP}_{\text{leaves}\leq l}$ reduces to $\text{IDPPH}_{\text{leaves}\leq l}$ via the identity mapping. Thus, the next theorem implies that $\text{IDPPH}_{\text{leaves}\leq l}$ is also NP-complete, which was previously proved by Gramm et al. [13] for $l = 2$.

Next, it is also easy to see that any directed problem reduces to the undirected version by adding an all-0-row. Thus, $\text{IPP}_{\text{leaves}\leq l}$ is also NP-complete. Indeed, all previously known NP-completeness results for variants of IPPH follow from Theorem 4.2, except for the NP-completeness of IPP.

Since IDPP $\in$ P, this is a first example of a perfect *path* phylogeny problem being harder than the corresponding problem for general perfect phylogenies. Our proof is based on a reduction from the NP-complete problem MONOTONE NAE3SAT and is similar to the reduction presented in [13], which starts, however, from NAE3SAT. By starting our reduction from a (conceptually) simpler problem we are able to prove a stronger result than the one presented in [13].

Theorem 4.2. $\text{IDPP}_{\text{leaves}\leq l}$ *is* NP-*complete for every* $l \geq 2$.

Proof. Fix an $l \geq 2$. We reduce MONOTONE NAE3SAT to $\text{IDPP}_{\text{leaves}\leq l}$. Let $\phi = \bigwedge_{i=1}^m C_i$ be an instance of MONOTONE NAE3SAT with m clauses $C_i = (x^i_0 \vee x^i_1 \vee x^i_2)$ and variables from $V = \{v_1,\dots,v_n\}$. We construct an incomplete $(n+3m+l-2) \times (3m+l-2)$ haplotype matrix B. The first n rows, which we call *variable rows*, are identified with the variables of ϕ. The next $3m$ rows and the first $3m$ columns are called *literal rows* and *literal columns*, respectively. They consist of rows c^i_j for $i \in \{1,\dots,m\}, j \in \{1,\dots,3\}$ and columns with the same names. The remaining $l-2$ columns are marked by $b_1, \dots, b_{l-2}$. First, we describe the non-?-entries of the upper left $(n+3m) \times (3m)$ submatrix: For each clause $C_i = (x^i_1 \vee x^i_2 \vee x^i_3)$ we put in each literal column c^i_j a 1-entry in variable row x^i_j. Then we put the submatrix $\left[\begin{smallmatrix}1&0&?\\0&?&1\\?&1&0\end{smallmatrix}\right]$ in columns c^i_1, c^i_2, and c^i_3 and rows c^i_1, c^i_2, and c^i_3. Finally, we set the lower right $(l-2) \times (l-2)$ submatrix to the identity matrix and all entries in the upper right $(n+3m) \times (l-2)$

submatrix and the lower left $(l-2) \times 3m$ submatrix to 0. An example of this construction for $l = 4$ is depicted in Figure 8.

$$\phi = C_1 \wedge C_2 \wedge C_3 \quad \text{with}$$

$$C_1 = (\underbrace{v_1}_{x_1^1}, \underbrace{v_2}_{x_2^1}, \underbrace{v_3}_{x_3^1}),$$

$$C_2 = (\underbrace{v_2}_{x_1^2}, \underbrace{v_3}_{x_2^2}, \underbrace{v_4}_{x_3^2}),$$

$$C_3 = (\underbrace{v_2}_{x_1^3}, \underbrace{v_4}_{x_2^3}, \underbrace{v_5}_{x_3^3})$$

is mapped to

$$\begin{array}{c|ccccccccc|cc|}
 & c_1^1 & c_2^1 & c_3^1 & c_1^2 & c_2^2 & c_3^2 & c_1^3 & c_2^3 & c_3^3 & b_1 & b_2 \\
\hline
v_1 & 1 & & & & & & & & & & \\
v_2 & & 1 & & 1 & & & 1 & & & & \\
v_3 & & & 1 & & 1 & & & & & & \\
v_4 & & & & & & 1 & & 1 & & & \\
v_5 & & & & & & & & & 1 & & \\
c_1^1 & 1 & 0 & ? & & & & & & & & \\
c_2^1 & 0 & ? & 1 & & & & & ? & & 0 & \\
c_3^1 & ? & 1 & 0 & & & & & & & & \\
c_1^2 & & & & 1 & 0 & ? & & & & & \\
c_2^2 & & & & 0 & ? & 1 & & & & & \\
c_3^2 & & & & ? & 1 & 0 & & & & & \\
c_1^3 & & & & & & & 1 & 0 & ? & & \\
c_2^3 & & ? & & & & & 0 & ? & 1 & & \\
c_3^3 & & & & & & & ? & 1 & 0 & & \\
\hline
 & & & & & 0 & & & & & 1 & 0 \\
 & & & & & & & & & & 0 & 1 \\
\hline
\end{array}$$

Figure 8: Example of the reduction from MONOTONE NAE3SAT to $\text{IDPP}_{\text{leaves}\le 4}$.

It remains to prove that $\phi \in$ MONOTONE NAE3SAT if, and only if, $B \in \text{IDPP}_{\text{leaves}\le l}$.

First, let B' be a completion of B that admits a directed perfect phylogeny T with at most l leaves. Our first claim is that the columns $b_1, \dots, b_{l-2}$ lie on $l-2$ different branches, each of which contains only this one column b_i. To see this, just note that for any column b_i, no other column of B' can lie on the same root-to-leaf path as b_i since these columns contain the submatrix $\left[\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right]$. The remaining columns, which are the literal columns, must then lie on at most two further branches T_0 and T_1 (recall that T has at most l leaves). Both branches must be nonempty since B contains in its literal columns the submatrix $\left[\begin{smallmatrix}1&0&?\\0&?&1\\?&1&0\end{smallmatrix}\right]$, whose missing entries, due to Lemma 2.2, must be filled with at least two 1-entries and therefore its columns lie on at least two different branches. The first n rows assure that all literal columns that correspond to the same variable lie on the same path: they contain 1-entries in the same row. The next $3m$ rows enforce that the literal columns of any given clause do not all lie on the same root-to-leaf path. We can now construct the desired truth assignment τ for the variables of ϕ that witnesses $\phi \in$ MONOTONE NAE3SAT: For a variable v_i, we set $\tau(v_i) = 0$, if the corresponding literal columns lie on T_0, and $\tau(v_i) = 1$, if they lie on T_1.

For the other direction let $\tau : \{v_1, \dots, v_n\} \to \{0,1\}$ be an assignment of the variables of ϕ such that the literals of a clause do not all share the same truth value. We describe, simultaneously, a completion B' for B and a directed perfect phylogeny T with at most l leaves for B'. First, T contains $l-2$ branches, each of which contains exactly one column b_i. The leaves at the ends of these branches are labeled with one of the lower $l-2$ rows of B. Next, there are two further paths T_0 and T_1 in T and these paths contain the completions of all of the remaining columns.

The path T_0 contains all literal columns whose corresponding variables are set to 0 by τ. The ordering of the columns on the path in root-to-leaf order is as follows: First, all literal columns for a clause C_i come earlier than all literal columns for clause C_{i+1}. For a single clause $C_i = (x_1^i \vee x_2^i \vee x_3^i)$, we only need to explain what happens when there are exactly two x_j^i and x_k^i inside C_i with $\tau(x_j^i) = \tau(x_k^i) = 0$. In this case, we may assume that $k \equiv j+1 \pmod 3$ holds (otherwise exchange the meanings of j and k). Then column c_j^i comes before c_k^i on the path T_0. Each edge on the path T_0 is now labeled with some column c_j^i. We label the node following this column by the row c_j^i; note that this positioning implicitly yields a completion for this row. The branch T_1 is constructed in the same way, but we now consider only variables with $\tau(x_j^i) = 1$. The last node on the path T_0 is labeled by all variable rows v_i

with $\tau(v_i) = 0$, similarly for T_1 and variables with $\tau(v_i) = 1$. Again, the positioning implicitly assigns completions to these rows. This completes the construction of the completion B' and of the sought directed perfect phylogeny with at most l leaves. □

5 Conclusion

We have shown that restrictions on the topologies of perfect phylogenies can greatly influence the complexity of IPPH and its variants. While restrictions can make the complexity jump from P to NP-complete (as for IDPP), we showed that tree topologies provide the first parameter for which a theoretical analysis is possible of an algorithm that works on arbitrary instances of the IPPH problem. Our new notions of mutation trees and light and heavy component sizes have turned out to be useful in the study of undirected perfect phylogenies; we suggest trying to apply them to other problem versions as well.

The main open problem is to improve the runtime of the fixed-parameter algorithm since the runtime is in the range of $3^{O(k!)}$, which is not feasible even for small values like $k = 5$ that are common in practice. One could argue that, in practice, the algorithm will be much quicker because the bound is only a rather pessimistic worst-case bound, but a faster fixed-parameter algorithm would be a much better alternative.

References

[1] V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph. Haplotyping as perfect phylogeny: A direct approach. *Journal of Computational Biology*, 10(3–4):323–340, 2003. Available from: http://dx.doi.org/10.1089/10665270360688048.

[2] C. J. Benham, S. Kannan, M. Paterson, and T. Warnow. Hen's teeth and whale's feet: Generalized characters and their compatibility. *Journal of Computational Biology*, 2(4):515–525, 1995. Available from: http://dx.doi.org/10.1089/cmb.1995.2.515.

[3] P. Bonizzoni. A linear-time algorithm for the perfect phylogeny haplotype problem. *Algorithmica*, 48(3):267–285, 2007. Available from: http://dx.doi.org/10.1007/s00453-007-0094-3.

[4] A. G. Clark. Inference of haplotypes from PCR-amplified samples of diploid populations. *Journal of Molecular Biology and Evolution*, 7(2):111–122, 1990. Available from: http://mbe.oxfordjournals.org/content/7/2/111.abstract.

[5] Z. Ding, V. Filkov, and D. Gusfield. A linear-time algorithm for the perfect phylogeny haplotyping (PPH) problem. *Journal of Computational Biology*, 13(2):522–553, 2006. Available from: http://dx.doi.org/10.1089/cmb.2006.13.522.

[6] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer-Verlag, New York, 1999.

[7] M. Elberfeld. Perfect phylogeny haplotyping is complete for logspace. *Computing Research Repository (CoRR)*, abs/0905.0602, 2009. Available from: http://arxiv.org/abs/0905.0602.

[8] M. Elberfeld, I. Schnoor, and T. Tantau. Influence of tree topology restrictions on the complexity of haplotyping with missing data. In *Proceedings of the 6th Annual Conference on Theory and Applications of Models of Computation (TAMC 2009)*, volume 5532 of *Lecture Notes in Computer Science*, pages 201–210. Springer, 2009. Available from: http://dx.doi.org/10.1007/978-3-642-02017-9_23.

[9] M. Elberfeld and T. Tantau. Computational complexity of perfect-phylogeny-related haplotyping problems. In *Proceedings of the 33rd International Symposium on Mathematical Foundations of Computer Science (MFCS 2008)*, volume 5162 of *Lecture Notes in Computer Science*, pages 299–310. Springer, 2008. Available from: http://dx.doi.org/10.1007/978-3-540-85238-4_24.

[10] E. Eskin, E. Halperin, and R. M. Karp. Efficient reconstruction of haplotype structure via perfect phylogeny. *Journal of Bioinformatics and Computational Biology*, 1(1):1–20, 2003. Available from: http://dx.doi.org/10.1142/S0219720003000174.

[11] L. Excoffier and M. Slatkin. Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Molecular Biology and Evolution*, 12(5):921–7, 1995. Available from: http://mbe.oxfordjournals.org/cgi/content/abstract/12/5/921.

[12] J. Gramm, T. Hartman, T. Nierhoff, R. Sharan, and T. Tantau. On the complexity of snp block partitioning under the perfect phylogeny model. *Discrete Mathematics*, 309(18):5610–5617, 2009. Available from: http://dx.doi.org/10.1016/j.disc.2008.04.002.

[13] J. Gramm, T. Nierhoff, R. Sharan, and T. Tantau. Haplotyping with missing data via perfect path phylogenies. *Discrete and Applied Mathematics*, 155(6–7):788–805, 2007. Available from: http://dx.doi.org/10.1016/j.dam.2005.09.020.

[14] D. Gusfield. Inference of haplotypes from samples of diploid populations: Complexity and algorithms. *Journal of Computational Biology*, 8(3):305–23, 2001. Available from: http://dx.doi.org/10.1089/10665270152530863.

[15] D. Gusfield. Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. In *Proceedings of the 6th Annual International Conference on Computational Biology (RECOMB 2002)*, pages 166–175. ACM Press, 2002. Available from: http://dx.doi.org/10.1145/565196.565218.

[16] E. Halperin and R. M. Karp. Perfect phylogeny and haplotype assignment. In *Proceedings of the 8th Annual International Conference on Computational Biology (RECOMB 2004)*, pages 10–19. ACM Press, 2004. Available from: http://dx.doi.org/10.1145/974614.974617.

[17] G. Kimmel and R. Shamir. The incomplete perfect phylogeny haplotype problem. *Journal of Bioinformatics and Computational Biology*, 3(2):359–384, 2005. Available from: http://dx.doi.org/10.1142/S0219720005001090.

[18] Y. Liu and C.-Q. Zhang. A linear solution for haplotype perfect phylogeny problem. In *Proceedings of the International Conference on Bioinformatics and its Applications*, pages 173–184. World Scientific, 2005. Available from: http://dx.doi.org/10.1142/9789812702098_0016.

[19] I. Pe'er, T. Pupko, R. Shamir, and R. Sharan. Incomplete directed perfect phylogeny. *SIAM Journal on Computing*, 33(3):590–607, 2004. Available from: http://dx.doi.org/10.1137/S0097539702406510.

[20] R. V. Satya and A. Mukherjee. An optimal algorithm for perfect phylogeny haplotyping. *Journal of Computational Biology*, 13(4):897–928, 2006. Available from: http://dx.doi.org/10.1089/cmb.2006.13.897.

[21] R. V. Satya and A. Mukherjee. The undirected incomplete perfect phylogeny problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 5(4):618–629, 2008. Available from: http://dx.doi.org/10.1109/TCBB.2007.70218.

[22] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing (STOC 2010)*, pages 216–226. ACM Press, 1978. Available from: http://dx.doi.org/10.1145/800133.804350.

[23] M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9(1):91–116, 1992. Available from: http://dx.doi.org/10.1007/BF02618470.

[24] J. Zhang, W. L. Rowe, A. G. Clark, and K. H. Buetow. Genomewide distribution of high-frequency, completely mismatching SNP haplotype pairs observed to be common across human populations. *American Journal of Human Genetics*, 73(5):1073–81, 2003. Available from: http://dx.doi.org/10.1086/379154.